6.5 COMPUTATION OF DISCRETE QUADRATIC TFDs [0]

6.5.1 General Computational Procedure

Article 6.1 deals with definitions and properties of the discrete WVD (DWVD) and other discrete quadratic TFDs. It shows that the general discrete quadratic TFD of an analytic signal $z[n]$ is

$$\rho_z[n,k] \;=\; 2 \sum_{|m|<\frac{M}{2}} \sum_{|p|<\frac{P}{2}} G[p,m]\, z[n-p+m]\, z^*[n-p-m]\, e^{-j2\pi km/M} \qquad (6.5.1)$$

$$\;=\; 2 \operatorname*{DFT}_{m\to k} \left\{ G[n,m] \underset{n}{*} \left(z[n+m]\, z^*[n-m] \right) \right\};\; m \in \langle M \rangle \qquad (6.5.2)$$

where the support dimensions of the kernel do not exceed M samples in the lag (m) direction and P samples in the time (n) direction, and $\langle M \rangle$ means any set of M consecutive integers; cf. [1, p. 444]. So the general procedure for evaluating such a TFD is:

1. Formation of the instantaneous autocorrelation function (IAF) $K_z[n,m] = z[n+m]\, z^*[n-m]$;
2. Discrete convolution in n (time) with the smoothing function $G[n,m]$;
3. Discrete Fourier transformation mapping m (lag) to k (frequency).

For the DWVD, which has $G[n,m] = \delta[n]$, step 2 reduces to an identity transformation and may be omitted. The windowed DWVD has $G[n,m] = \delta[n]\, g[m]$, so that step 2 reduces to multiplication of the IAF by $g[m]$. Some other quadratic TFDs, however, have special forms leading to computational procedures which are *not* degenerate cases of the above, and which may be simpler or faster.

This article addresses some of the practical issues in computing quadratic TFDs of a real signal, examines various cases of the above procedure, and considers the spectrogram as one example of a special form leading to a simpler, faster algorithm.

6.5.2 Computation of the Analytic Signal

The usual definitions of quadratic TFDs, especially the WVD and the windowed WVD, assume an analytic signal in order to avoid interference terms between positive and negative frequencies. For computational purposes, an analytic signal also avoids the need for $2\times$ oversampling prior to computation of the IAF (see Section 6.1.1 and ref. [2]). So, given a real signal $s(t)$, we must first compute the analytic signal $z(t)$ associated with $s(t)$. The simplest method is the direct approach of filtering out the negative frequencies in the frequency domain. If a real signal $s[n]$ is given for $n = 0, 1, 2, \ldots, N-1$ and periodically extended with period N, where N is even (or is made even by zero-padding), the algorithm is:

[0] Authors: **Boualem Boashash** and **Gavin R. Putland**, Signal Processing Research Centre, Queensland University of Technology, GPO Box 2434, Brisbane, Q 4001, Australia (b.boashash@qut.edu.au, g.putland@qut.edu.au). Reviewers: S. L. Marple, A. Reilly and V. Sucic.

1. Compute $S[k] = \mathrm{DFT}\{s[n]\}$ for $k = 0, 1, \ldots, N-1$;
2. Compute
$$Z[k] = \begin{cases} S[k] & \text{for } k = 0, \frac{N}{2} \\ 2\,S[k] & \text{for } k = 1, 2, \ldots, \frac{N}{2}-1 \\ 0 & \text{otherwise ;} \end{cases} \tag{6.5.3}$$
3. Compute $z[n] = \mathrm{IDFT}\{Z[k]\}$, where $\mathrm{IDFT}\{\ldots\}$ denotes the inverse DFT.

The treatment of the Nyquist term ($k = N/2$) and the precise meaning of "analytic" for a discrete-time periodic signal are explained in [3]; these issues become significant if the signal has non-zero amplitude at the Nyquist frequency. Further details on implementation of TFDs, including computation of the analytic signal, are given in [4]. A time-domain algorithm for computing the analytic signal using FIR filters is described in [5].

6.5.3 Real-Time Computation of TFDs

The formula for the discrete quadratic TFD [Eq. (6.5.2)] involves the expression $z[n+m]$ where m is allowed to be positive, together with $z^*[n-m]$ where m is allowed to be negative. The same applies to the DWVD

$$W_z[n,k] = 2 \underset{m \to k}{\mathrm{DFT}} \{z[n+m]\, z^*[n-m]\} \;; \quad m \in \langle N \rangle \tag{6.5.4}$$

and the windowed DWVD

$$W_z^g[n,k] = 2 \underset{m \to k}{\mathrm{DFT}} \{g[m]\, z[n+m]\, z^*[n-m]\} \;; \quad m \in \langle M \rangle \tag{6.5.5}$$

(both of these expressions are derived in Article 6.1). Both cases involve **time-advanced** signals; for any value of n, the computation of the TFD involves signal values up to $z[n+\lambda]$, where λ is some positive integer. In real-time computation, we cannot compute the TFD for time n until we know the signal values up to $z[n+\lambda]$; thus λ is the minimum **latency** of the computation. In the case of the DWVD [Eq. (6.5.4)], the range of m for which the IAF can be non-zero is maximized when n is at the center of the time-support of the signal; so the latency reaches a peak of half the signal duration. For the windowed DWVD, the latency is limited to half the window duration. For the general discrete quadratic TFD, the latency is limited to half the sum of the dimensions of the G matrix. The latency of the analytic signal computation must be added to that of the TFD computation. In all cases a smaller value of M not only reduces latency but also produces shorter FFTs, hence shorter computational delays; but the cost is reduced frequency resolution.

Latency is one of two measures of merit for real-time computation of TFDs. The other measure is throughput, which depends on the efficiency of the numerical algorithms. Eq. (6.5.2) can be written

$$\rho_z[n,k] = 2 \underset{m \to k}{\mathrm{DFT}} \{R_z[n,m]\} \tag{6.5.6}$$

where $R_z[n,m] = G[n,m] \underset{n}{*} (z[n+m]\, z^*[n-m])$. Similarly,

$$\rho_z[n+1,k] = 2 \underset{m\to k}{\mathrm{DFT}} \{R_z[n+1,m]\}. \tag{6.5.7}$$

The above two equations represent successive time-slices of the TFD. Multiplying the second equation by j, adding the result to the first equation and using the linearity of the DFT, we obtain

$$\rho_z[n,k] + j\rho_z[n+1,k] = 2 \underset{m\to k}{\mathrm{DFT}} \{R_z[n,m] + jR_z[n+1,m]\}. \tag{6.5.8}$$

If the TFD is known *a priori* to be real, as it usually is, then Eq. (6.5.8) means that the successive time slices of the TFD are respectively the real and imaginary parts of the right-hand side, which involves only one FFT [6]. Thus the realness property can enhance efficiency by halving the required number of FFTs. It can also halve the storage requirement as it implies Hermitian symmetry in the smoothed IAF.

6.5.4 Computational Approximations for Discrete-Time Kernels

Table 6.5.1 reproduces the "$G[n,m]$" column of Table 6.1.2 (p. 240) and adds two special cases often found in the literature: $\mathrm{BJ}_{1/2}$ denotes the Born-Jordan distribution with $\alpha = 1/2$, while ZAM_2 denotes the Zhao-Atlas-Marks distribution with $a = 2$. Some entries in the "$G[n,m]$" column of Table 6.5.1 call for continuous convolution prior to sampling. At best, the evaluation of such a convolution in the time-lag domain requires oversampling. At worst, it requires the numerical evaluation of an improper integral arising from a singularity in $G(t,\tau)$. In either case, computational inefficiencies will arise unless the smoothing effect of the convolution can be approximated in some other way. The chosen approximations, shown in the right-hand column of Table 6.5.1, are explained below.

In the case of the B-distribution, the sole purpose of the convolution is to avoid aliasing. Without the convolution, and for typical values of the parameter β (e.g. $\beta = 0.01$), the time-lag kernel would be a continuous function with a narrow slot at $m = 0$ caused by the factor $|2m|^\beta$. This factor is approximately unity for small nonzero values of m. The convolution fills in the slot, so that the factor is approximately unity at $m = 0$ also. This effect can be approximated by replacing $|2m|$ with $[4m^2+1]^{1/2}$, as is done in Table 6.5.1.

In the case of ZAM distribution, for a suitable (unbounded) $w[m]$, the convolution also ensures that $G[n,0] = \delta[n]$, which in turn verifies the TM property. Without the convolution, we would have

$$G_{\mathrm{ZAM}}[n,m] = w[m]\,\mathrm{rect}\big(\tfrac{an}{4m}\big) = \begin{cases} w[m] & \text{if } |an| \le |2m| \\ 0 & \text{otherwise.} \end{cases} \tag{6.5.9}$$

This gives $G[n,0] = w[0]\,\delta[n]$, which verifies the TM property provided that $w[0] = 1$. Accordingly, Eq. (6.5.9) is used in Table 6.5.1, although other approximations are possible. For example, we could sacrifice the TM property in favor of

Table 6.5.1: Computational approximations for time-lag kernels of selected discrete quadratic TFDs. In the "Distribution" column, subscripts indicate parameter values while the prefix "w-" means "windowed" by the function $w[m]$. For the spectrogram and w-Levin distributions, the window w is assumed to be real and even. The "$G[n,m]$" column shows the exact kernels required for the avoidance of aliasing in the Doppler-frequency domain. If $G[n,m]$ cannot be computed as written, the "Approx." column shows the suggested computational approximation.

Distribution	$G[n,m]$	Approx.
WVD	$\delta[n]$	
Levin	$\frac{1}{2}\delta[n+m] + \frac{1}{2}\delta[n-m]$	
BJ	$\big[\frac{1}{\lvert 4\alpha m\rvert}\,\mathrm{rect}\big(\frac{n}{4\alpha m}\big)\big]$ $**\,[\,\mathrm{sinc}\, n\,\mathrm{sinc}\, m]$	$\approx \begin{cases} \frac{1}{\lvert 4\alpha m\rvert+1} & \text{if } \lvert 2n\rvert \le \lvert 4\alpha m\rvert + 1 \\ 0 & \text{otherwise.} \end{cases}$
$\mathrm{BJ}_{1/2}$	$\big[\frac{1}{\lvert 2m\rvert}\,\mathrm{rect}\big(\frac{n}{2m}\big)\big]$ $**\,[\,\mathrm{sinc}\, n\,\mathrm{sinc}\, m]$	$\approx \begin{cases} \frac{1}{\lvert 2m\rvert+1} & \text{if } \lvert n\rvert \le \lvert m\rvert \\ 0 & \text{otherwise.} \end{cases}$
Modified B	$\frac{\cosh^{-2\beta} n}{\sum_n \cosh^{-2\beta} n}$	
w-WVD	$\delta[n]\, w[m]$	
w-Levin	$\frac{1}{2}w[m]\,(\delta[n+m] + \delta[n-m])$	
ZAM	$\big[w[m]\,\mathrm{rect}\big(\frac{an}{4m}\big)\big]$ $**\,[\,\mathrm{sinc}\, n\,\mathrm{sinc}\, m]$	$\approx \begin{cases} w[m] & \text{if } \lvert an\rvert \le \lvert 2m\rvert \\ 0 & \text{otherwise.} \end{cases}$
ZAM_2	$\big[w[m]\,\mathrm{rect}\big(\frac{n}{2m}\big)\big]$ $**\,[\,\mathrm{sinc}\, n\,\mathrm{sinc}\, m]$	$\approx \begin{cases} w[m] & \text{if } \lvert n\rvert \le \lvert m\rvert \\ 0 & \text{otherwise.} \end{cases}$
Rihaczek	$\delta[n-m]$	
w-Rihaczek	$w^*[-m]\,\delta[n-m]$	
Page	$\delta[n-\lvert m\rvert]$	
CW	$\frac{\sqrt{\pi\sigma}}{\lvert 2m\rvert}\exp\Big(\frac{-\pi^2\sigma n^2}{4m^2}\Big)$ $**\,[\,\mathrm{sinc}\, n\,\mathrm{sinc}\, m]$	$\approx \begin{cases} \delta[n] & \text{if } m = 0 \\ \sqrt{\frac{\pi\sigma}{4m^2+\pi\sigma}}\exp\Big(\frac{-\pi^2\sigma n^2}{4m^2+\pi\sigma}\Big) & \text{otherwise.} \end{cases}$
B	$\Big[\frac{\lvert 2m\rvert}{\cosh^2 n}\Big]^{\beta} * \mathrm{sinc}\, m$	$\approx \Big[\frac{\sqrt{4m^2+1}}{\cosh^2 n}\Big]^{\beta}$
spectrogram	$w[n+m]\, w[n-m]$	

some smoothing by using the approximation

$$G_{\mathrm{ZAM}}[n,m] \approx \tfrac{1}{2}w[m]\big[1 + \tanh(\lvert 4m\rvert - \lvert 2an\rvert)\big]\ , \tag{6.5.10}$$

and we could salvage the TM property by using a separate definition for $m = 0$.

For the Born-Jordan (BJ) and Choi-Williams (CW) distributions, the convolutions are needed to remove singularities at $m = 0$ and ensure that $G[n, 0] = \delta[n]$. For the BJ distribution, we can remove the singularity and approximate the spreading in the $[n, m]$ plane by replacing $|4\alpha m|$ with $|4\alpha m|+1$. The result is

$$G_{\mathrm{BJ}}[n, m] \approx \tfrac{1}{|4\alpha m|+1} \operatorname{rect}\big(\tfrac{n}{|4\alpha m|+1}\big) \ , \tag{6.5.11}$$

which is equivalent to the rule given in Table 6.5.1. For the CW distribution, a similar effect is obtained by replacing $|2m|$ with $[4m^2+\pi\sigma]^{1/2}$. This step, by itself, gives the kernel

$$G_{\mathrm{CW}}[n, m] \approx \sqrt{\tfrac{\pi\sigma}{4m^2+\pi\sigma}} \ \exp\big(\tfrac{-\pi^2\sigma n^2}{4m^2+\pi\sigma}\big). \tag{6.5.12}$$

For $n = m = 0$, this reduces to $G[0, 0] = 1$, which is consistent with the requirement that $G[n, 0] = \delta[n]$. However, for $m = 0$, Eq. (6.5.12) reduces to $G[n, 0] = e^{-\pi n^2}$, which is only an approximation to $\delta[n]$. Accordingly, a two-part definition of the kernel is used in the "Approx." column of the table. With $n = 0$, the kernel as defined in the table reduces to

$$G_{\mathrm{CW}}[0, m] = \sqrt{\tfrac{\pi\sigma}{4m^2+\pi\sigma}} \tag{6.5.13}$$

which takes the value 1 at $m = 0$ and $1/\sqrt{2}$ at $m = \pm\sqrt{\pi\sigma/4}$. For realistic values of σ (e.g. $\sigma \geq 1$), this gives a reasonable degree of smoothing in the m direction.

An alternative approach to the problem of singularities, which is not pursued here, is to evaluate the kernels in the Doppler-lag $[l, m]$ domain. This is efficient if we intend to evaluate the time-convolution by the FFT method, which also uses the $[l, m]$ domain. But it is still an approximation (if the time-lag form of the kernel is taken as the definition) because the analytical formulae for continuous FTs of standard signals are only approximations when applied to the DFT.

6.5.5 Special Case: Direct Form of the Discrete Spectrogram

The short-time Fourier transform (STFT) of the continuous-time signal $x(t)$ with real window $w(t)$ is defined (in Section 2.3.1) as

$$F_x^w(t, f) \stackrel{\triangle}{=} \underset{\tau\to f}{\mathcal{F}}\{x(\tau)\, w(\tau - t)\} = e^{-j2\pi ft} \underset{\tau\to f}{\mathcal{F}}\{x(\tau + t)\, w(\tau)\} \tag{6.5.14}$$

$$= e^{-j2\pi ft} \int_{-\infty}^{\infty} x(\tau + t)\, w(\tau)\, e^{-j2\pi f\tau} d\tau. \tag{6.5.15}$$

It is shown in Chapter 2 that the spectrogram $S_x^w(t, f)$, which is simply the squared magnitude of the STFT, can also be considered as a quadratic TFD with kernel $w(t+\frac{\tau}{2})\, w(t-\frac{\tau}{2})$. The discrete form of this kernel is $w[n+m]\, w[n-m]$. Hence the discrete spectrogram can be conveniently evaluated using the general procedure described in Section 6.5.1 above. But it is simpler and more efficient to discretize the continuous spectrogram directly.

Theorem 6.5.1: If the spectrogram S_x^w is modified by ideally sampling $w(\tau)$ at

$$\tau = m/f_s \tag{6.5.16}$$

where m is an integer and f_s is the sampling rate, and if

$$w(\tau) = 0 \quad \text{for} \quad |\tau| \geq \tfrac{M}{2f_s} \tag{6.5.17}$$

where M is a positive integer, and if the modified TFD is denoted by $\widehat{S}_x^w$, then

$$\widehat{S}_x^w\big(\tfrac{n}{f_s}, \tfrac{kf_s}{M}\big) = \left| \sum_{|m|<M/2} x\big(\tfrac{m+n}{f_s}\big)\, w\big(\tfrac{m}{f_s}\big)\, e^{-j2\pi km/M} \right|^2. \tag{6.5.18}$$

Proof/explanation: When $w(\tau)$ is sampled, the integrand in Eq. (6.5.15) becomes

$$x(\tau+t)\, w(\tau)\, e^{-j2\pi f\tau} \sum_{m=-\infty}^{\infty} \delta\big(\tau - \tfrac{m}{f_s}\big) \tag{6.5.19}$$

so that the STFT becomes

$$\widehat{F}_x^w(t,f) = e^{-j2\pi ft} \sum_{m=-\infty}^{\infty} x\big(\tfrac{m}{f_s}+t\big)\, w\big(\tfrac{m}{f_s}\big)\, e^{-j2\pi fm/f_s}. \tag{6.5.20}$$

By Eqs. (6.5.16) and (6.5.17), the summation is restricted to $|m| < M/2$, giving a maximum of M terms.[1] The sampling in τ makes $\widehat{F}_x^w(t,f)$ periodic in f with period f_s, while the time-limiting in τ gives a frequency resolution of M bins per period. So it is convenient to let

$$f = kf_s/M \tag{6.5.21}$$

where k is an integer. With these restrictions, Eq. (6.5.20) becomes

$$\widehat{F}_x^w\big(t, \tfrac{kf_s}{M}\big) = e^{-j2\pi kf_s t/M} \sum_{|m|<M/2} x\big(\tfrac{m}{f_s}+t\big)\, w\big(\tfrac{m}{f_s}\big)\, e^{-j2\pi km/M}. \tag{6.5.22}$$

Putting $t = n/f_s$ to match the quantization of τ, then taking the squared magnitude of the discrete STFT, we obtain Eq. (6.5.18). ∎

With a change of notation, Eq. (6.5.18) becomes

$$S_x^w[n,k] = \left| \sum_{|m|<M/2} x[m+n]\, w[m]\, e^{-j2\pi km/M} \right|^2. \tag{6.5.23}$$

This $S_x^w[n,k]$ is the **discrete spectrogram** of the discrete-time signal $x[n]$ with window $w[m]$. If the summand is extended periodically in m with period M (i.e. extended periodically in τ with period M/f_s), we obtain

$$S_x^w[n,k] = \left| \sum_{m\in\langle M\rangle} x[m+n]\, w[m]\, e^{-j2\pi km/M} \right|^2 \tag{6.5.24}$$

[1] M terms for odd M; $M-1$ terms for even M.

where $\langle M \rangle$ denotes any set of M consecutive integers.[2] This may be written

$$S_x^w[n,k] = \Big| \underset{m \to k}{\mathrm{DFT}} \{x[m+n]\, w[m]\} \Big|^2 \;; \quad m \in \langle M \rangle \,. \tag{6.5.25}$$

The time support of $S_x^w[n,k]$ is that of $x[n] * w[n]$, corresponding to $x(t) * w(t)$. If this has a duration not exceeding N samples, then the non-zero elements of the discrete spectrogram may be represented by an $N \times M$ real matrix. Only half of the M columns are needed for the non-negative frequencies, which are sufficient if $x(t)$ is real.

Eq. (6.5.23) involves $x[n+m]$ where $|m| < M/2$ and M is the window length in samples. So, in real-time computations, the latency of the discrete spectrogram computed by this formula is half the window length.

6.5.6 Sample Code Fragments

In view of the current popularity of MATLAB™, we illustrate this Article with some code fragments from the experimental MATLAB function `tlkern.m`, which computed all of the TFDs plotted in Article 5.7. The input parameters of the function specify the kernel in terms of a time-dependent factor $g_1[n]$, a lag-dependent factor $g_2[m]$, and an "auxiliary factor" $g_3[n,m]$. The overall time-lag kernel $G[n,m]$ is then computed as

$$G[n,m] = g_2[m]\big(g_1[n] \underset{n}{*} g_3[n,m]\big) = \big(g_2[m]\, g_1[n]\big) \underset{n}{*} g_3[n,m]. \tag{6.5.26}$$

This scheme allows the computation of a wide variety of quadratic TFDs in under 320 lines of code, including exceptions for direct computation of the discrete spectrogram.

6.5.6.1 Example 1: MBD (General Algorithm)

For a separable kernel, the auxiliary factor would normally be omitted (i.e. taken as $\delta[n,m]$), while the time-dependent and lag-dependent factors would have input parameters specifying their types (e.g. Hamming or Hanning) and durations (in samples). Although the kernel of the modified B-distribution (MBD) is separable (see Article 5.7 and Table 6.5.1), its parameter β is *not* a duration. The MBD kernel is therefore specified using the factors

$$g_1[n] = \delta[n] \;; \quad g_2[m] = 1 \;; \quad g_3[n,m] = \tfrac{\cosh^{-2\beta} n}{\sum_n \cosh^{-2\beta} n} \,. \tag{6.5.27}$$

Notice that the auxiliary factor is the complete kernel.

To compute the MBD in Fig. 5.7.2(e) on p. 220, the function `tlkern` is called with the following significant parameters:

[2] For even M, the periodic extension is padded with a zero term.

`s` = signal vector
`N` = 128 = assumed period
`tr` = 1 = time resolution
`tf` = '`delta`' = string specifying $g_1[n]$
`lf` = '`1`' = string specifying $g_2[m]$
`af` = '`mb`' = string specifying form of $g_3[n, m]$
`ap` = 0.2 = auxiliary parameter (β).

All internal computations, including IAF generation, are designed to be valid for periodic signals. Therefore, to compute the IAF of a *non*-periodic signal such as the one in Fig. 5.7.2(e), the assumed period `N` must be at least twice the signal length to avoid wrap-around effects. Because the time support of the IAF is identical to that of the signal, the same value of `N` is also sufficient to avoid wrap-around in the subsequent convolution with $g_1[n]$.

The output is the real matrix `tfd(1:Mpad+1,1:Nsel)`, whose dimensions `Mpad+1` and `Nsel` are assigned early by the statements

```
Mpad = 2^ceil(log(2*M)/log(2));  % lag-to-frequency FFT length
Ncut = min(N,length(s));         % duration of TF plot
Nsel = ceil(Ncut/tr);            % no. traces in TF plot
```

where `M` is the support length of the kernel in the lag direction; in this case `M` has been set to `length(s)` because of the constant "lag-dependent" factor.

Preliminaries: The analytic signal is computed by the frequency-domain method. If `N` is even, the Nyquist term has MATLAB index `N/2+1` and the amplitude at that frequency is left unchanged [3]. If `N` is odd, there is no Nyquist term. The following code handles both cases:

```
Noff = fix(N/2);
z = fft(real(s),N);           % s truncated or padded
z(2:N-Noff) = 2*z(2:N-Noff);  % positive frequencies
z(Noff+2:N) = 0;              % negative frequencies
z = ifft(z);
```

For this kernel, the time-dependent factor `g1` and the lag-dependent factor `g2` are computed by the statements

```
g1(1:N) = 0;
...
g1(1) = 1;
...
g2(1:Mpad) = 1;
```

where "`...`" denotes one or more line(s) of control code, or code that is skipped in this case. The auxiliary factor `g3` (the whole kernel in this case) is computed by

```
Moff = fix(M/2);
...
g3(1:N,1:Mpad) = 0;
...
```

```
temp(1:N) = 0;
for n = -Noff:Noff
  temp(1+rem(N+n,N)) = (cosh(n))^(-2*ap);
end
temp = temp/sum(temp); % normalize
for m = -Moff:Moff
  g3(:,1+rem(Mpad+m,Mpad)) = temp.';
end
```

where `ap` denotes the auxiliary parameter (β), and the remainder (`rem`) function causes high array indices to represent negative values of time and lag.

Step 1—Formation of the IAF: The IAF matrix `K(1:N,1:Mpad)` is formed by

```
for n = 1:N
  for m = -Moff:Moff
  % K(n,m) = z(n+m)z^*(n-m), with corrected indices:
    K(n,1+rem(Mpad+m,Mpad)) = z(1+rem(2*N+n+m-1,N))*conj(z(1+rem(2*N+n-m-1,N)));
  end
end
```

where the "corrected" indices allow handling of periodic signals.

Step 2—Convolution in time: The assembly of the time-lag kernel and the convolution in time with the IAF are performed together. The smoothed IAF is

$$R_z[n,m] = K_z[n,m] \underset{n}{*} G[n,m] = K_z[n,m] \underset{n}{*} \big(g_2[m]\, g_1[n] \underset{n}{*} g_3[n,m]\big). \qquad (6.5.28)$$

The above convolutions may be taken as circular if the assumed period is sufficiently long, in which case

$$R_z[n,m] = \underset{n\leftarrow l}{\mathrm{IDFT}} \left\{ \underset{n\to l}{\mathrm{DFT}} \{K_z[n,m]\} \underset{n\to l}{\mathrm{DFT}} \{g_3[n,m]\} \underset{n\to l}{\mathrm{DFT}} \{g_1[n]\, g_2[m]\} \right\}. \qquad (6.5.29)$$

This is implemented by the following code, in which `K(:,mcorr)` is initially the $\mathtt{m}^{\mathrm{th}}$ column of the IAF, but is overwritten by the $\mathtt{m}^{\mathrm{th}}$ column of the *smoothed* IAF:

```
for m = -Moff:Moff
  mcorr = 1+rem(Mpad+m,Mpad);
  ...
  K(:,mcorr) = ifft(fft(K(:,mcorr)).*fft(g3(:,mcorr)).*fft(g1.'*g2(mcorr)));
  ...
end
```

(The factor $g_2[m]$ could be taken outside the IDFT, but this would not improve the efficiency of the code because `g2(mcorr)` is a scalar.) The FFT method of convolution is useful in *experimental* code because of its generality, but is not necessarily the most efficient method, especially if one of the convolved sequences is short.

Now we apply the time-resolution (`tr`):

```
for nsel = 1:Nsel
% nselth column of r is selected row of K:
  n = 1+tr*(nsel-1);
  r(:,nsel) = K(n,:).';
end
```

Step 3—DFT: The final DFT (lag to frequency) is computed by

```
r = fft(r);
```

which, for the sake of generality, does *not* take advantage of realness.

Final adjustments: The following code scales the TFD and repeats its first row (the zero-frequency row) so that the TFD spans a full cycle in the frequency domain:

```
tfd = [real(r);real(r(1,:))].*(Ncut/Nsel/Mpad);
```

The scaling ensures that the sum of the matrix elements is close to the signal energy regardless of the time resolution.

6.5.6.2 Example 2: Spectrogram (Special Case)

The spectrogram in Fig. 5.7.3(f) on p. 221 was computed by the same function `tlkern`. For the spectrogram, the parameters `s`, `N` and `tr` are the same as for the MBD, while `tf` is ignored. Other significant parameters are

```
lf = 'rect' = string specifying type of window
 M = 17 = window length (in samples)
af = 'sg' = string calling for spectrogram.
```

The output is `tfd(1:Mpad/2+1,1:Nsel)`, where the dimensions are assigned as for the MBD, except that the window duration `M` is read as an input parameter and *not* overwritten.

The analytic signal is computed as for the MBD, although this is not strictly necessary for the spectrogram.

The rectangular window is computed by

```
Moff = fix(M/2);
g2(1:Mpad) = 0;
...
for m = -Moff:Moff
  g2(1+rem(Mpad+m,Mpad)) = 1;
end
```

The matrix `K(1:N,1:Mpad)` normally represents the IAF, but for the spectrogram it is assigned differently:

```
for n = 1:N
  for m = -Moff:Moff
  % K(n,m) = z(n+m)g2(m), with corrected indices:
    K(n,1+rem(Mpad+m,Mpad)) = z(1+rem(2*N+n+m-1,N))*g2(1+rem(Mpad+m,Mpad));
  end
end
```

The code that applies the time resolution and performs the final DFT (lag to frequency) is the same as for the MBD. But the final adjustment is different:

```
tfd = (abs(r(1:Mpad/2+1,:))).^2.*(Ncut/Nsel/Mpad/sum(g2.^2));
```

The magnitude-squared operation alters the relationship between the window and the scaling of the TFD. Also note that the above step uses only half the columns of the Fourier-transformed `r` matrix, namely those corresponding to the non-negative frequencies. Efficiency could be further improved by exploiting the analytic signal to halve the sampling rate.

6.5.7 The *TFSA* package

The *Time-Frequency Signal Analysis* (*TFSA*) package is a set of functions developed over more than a decade at the Signal Processing Research Centre, Queensland University of Technology, for computing modulated signals, quadratic and polynomial TFDs, ambiguity functions, wavelet transforms and scalograms, and various estimates of instantaneous frequency. As this is a production package rather than an experimental package, computationally intensive functions are precompiled and optimized for efficiency, and an interactive user interface is added. The current version is distributed as a MATLAB toolbox, so that TFSA functions can be used with other computational and graphical functions of MATLAB. Further information is available at http://www.sprc.qut.edu.au/ or http://www.eese.bee.qut.edu.au/research/spr/ .

6.5.8 Summary and Conclusions

High-level programming languages with built-in FFT functions and matrix operations have made it possible to construct compact yet highly versatile functions for computing quadratic TFDs. Use of a common algorithm for all TFDs is convenient for the programmer. But, as illustrated by the direct form of the spectrogram, efficiency can sometimes be improved by using different algorithms in special cases.

References

[1] B. Boashash, "Time-frequency signal analysis," in *Advances in Spectrum Analysis and Array Processing* (S. Haykin, ed.), vol. 1, ch. 9, pp. 418–517, Englewood Cliffs, NJ: Prentice-Hall, 1991.

[2] B. Boashash, "Note on the use of the Wigner distribution for time-frequency signal analysis," *IEEE Trans. Acoustics, Speech, & Signal Processing*, vol. 36, pp. 1518–1521, September 1988.

[3] S. L. Marple Jr., "Computing the discrete-time "analytic" signal via FFT," *IEEE Trans. Signal Processing*, vol. 47, pp. 2600–2603, September 1999.

[4] B. Boashash and A. Reilly, "Algorithms for time-frequency signal analysis," in *Time-Frequency Signal Analysis: Methods and Applications* (B. Boashash, ed.), ch. 7, pp. 163–181, Melbourne/N.Y.: Longman-Cheshire/Wiley, 1992.

[5] A. Reilly, G. Frazer, and B. Boashash, "Analytic signal generation—Tips and traps," *IEEE Trans. Signal Processing*, vol. 42, pp. 3241–3245, November 1994.

[6] B. Boashash and P. J. Black, "An efficient real-time implementation of the Wigner-Ville distribution," *IEEE Trans. Acoustics, Speech, & Signal Processing*, vol. 35, pp. 1611–1618, November 1987.